学号:031602523姓名:刘宏岩 学院:数计学院 专业:计算机类 班级:实验班

**《Linux 操作系统设计实践》**

**实验六:综合设计**

**实验环境:Ubuntu16.04**

**实验内容: 贪吃蛇**

**Github链接：https://github.com/liuhycn/Linux-ex6**

## 1. 相关API学习

#### 1.1GTK相关API

**opgtk\_window\_new(GTK\_WINDOW\_TOPLEVEL)**

* 函数创建一个窗口并返回这个窗口的控件指针。
* 参数 GTK\_WINDOW\_TOPLEVEL 指明窗口的类型为最上层的主窗口，它最常 用。还可以取另一个值 GTK\_WINDOW\_POPUP 指明窗口的类型为弹出式的无边框 的窗口。

**g\_signal\_connect()**

* 使用这个宏为窗口或控件加回调函数。  
  g\_signal\_connect 宏有 4 个参数，分别是: 连接的对象，就是要连接信号的控件的指针(**注意:必须是已创建完的控件的指针**)，需要用 G\_OBJECT 宏来转换; 信号名称，就是要连接的信号名称，为字符串形式，用双引号引起来。不 同的控件拥有的信号名称是不一样的;
* 回调函数，指信号发生时调用的函数，这里只用到函数名称，需要用G\_CALLBACK 宏来转换一下;传递给回调函数的参数，它的值类型应该为 gpointer。如果不是这一类型 可以强制转换，如果没有参数则为 NULL。这里只能传递一个参数，如果有多 个参数，可以先将多个参数定义为一个结构，再将此结构作为参数传递过来

**gtk\_window\_set\_title(window,const gchar\* title)**

* 设定窗口的标题:

**gtk\_window\_set\_default\_size(window,int width,int height)**

* 设定窗口的默认宽高

**gtk\_window\_set\_position(window,GtkWindowPosition**

**position);**

* 设定窗口的位置:
* 其中 position 可以取如下值

GTK\_WIN\_POS\_NONE 不固定  
GTK\_WIN\_POS\_CENTER 居中  
GTK\_WIN\_POS\_MOUSE 出现在鼠标位置 GTK\_WIN\_POS\_CENTER\_ALWAYS 窗口改变尺寸仍居中 GTK\_WIN\_POS\_CENTER\_ON\_PARENT 居于父窗口的中部

**gtk\_container\_add ()**

* 功能是将另一控件加入到容器中来。
* 它的第一参数是 GtkContainer 型的指针，这就需要将窗口控件指针用宏 GTK\_CONTAINER 转换一下，即 GTK\_CONTAINER(window)。它的第二参数是要容 纳的控件的指针，即 button。

**gtk\_widget\_show\_all()**

* 原本每一个控件都要用函数 gtk\_widget\_show 来显示，而这个函数显示容

器中所有控件。

* 它的参数是一个容器控件的指针，例如本次实验中用到的:

gtk\_widget\_show\_all(window);

#### 1.2文件读写相关API

**void open(const char\* filename,int mode,int access);**

* 打开文件，在fstream类中，有一个成员函数open()，就是用来打开文件的，其原型是：
* imbue(locale("chs"));设置中文模式
* 参数：

filename：　　 要打开的文件名  
mode：　　　 要打开文件的方式  
access：　　　 打开文件的属性  
打开文件的方式在类ios(是所有流式I/O类的基类)中定义，常用的值如下：  
ios::binary： 　以二进制方式打开文件，缺省的方式是文本方式。

ios::in：　　　 文件以输入方式打开  
ios::out：　　　文件以输出方式打开

**fstream.close()**

* 关闭文件，打开的文件使用完成后一定要关闭，fstream提供了成员函数来完成此操作，如：file1.close();就把file1相连的文件关闭。

**fstream.get()**

* 读取文件，get函数的用法是，从文本中一个一个字符的读入到程序运行的内存中，每读一次，就自动跳到文本的下一个字符

#### 1.3 Socket通信API

**int socket(int domain, int type, int protocol);**

* socket该函数用来创建一个套接字，并返回一个描述符，该描述符可以用来访问该套接字，函数中的三个参数分别对应前面所说的三个套接字属性protocol参数设置为0表示使用默认协议

**int bind( int socket, const struct sockaddr \*address, size\_t address\_len);**

* 该函数把通过socket调用创建的套接字命名，从而让它可以被其他进程使用。对于AF\_UNIX，调用该函数后套接字就会关联到一个文件系统路径名，对于AF\_INET，则会关联到一个IP端口号。成功时返回0，失败时返回-1；

**int sendto(int sockfd, void \*buffer, size\_t len, int flags, struct sockaddr \*to, socklen\_t tolen);**

* 该函数把缓冲区buffer中的信息给送给指定的IP端口的程序，buffer中储存着将要发送的数据，len是buffer的长度，而flags在应用中通常被设置为0，to是要发送数据到的程序的IP端口，tolen是to参数的长度。成功时返回发送的数据的字节数，失败时返回-1.

**int recvfrom(int sockfd, void \*buffer, size\_t len，int flags, struct sockaddr \*src\_from, socklen\_t \*src\_len);**

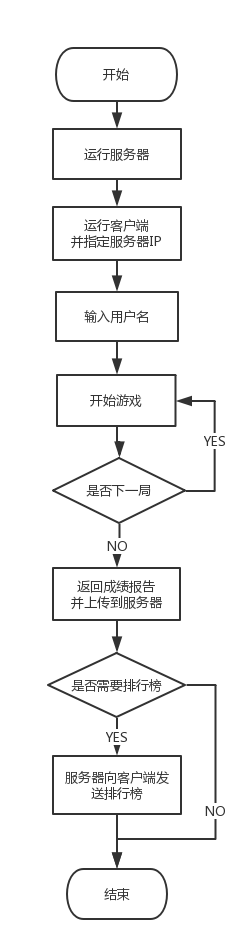
* 该函数把发送给程序的信息储存在缓冲区buffer中，并记录数据来源的程序IP端口，buffer用于储存接收到的数据，len指定buffer的长度，而flags在应用中通常被设置0，src\_from若不为空，则记录数据来源程序的IP端口，若src\_len不为空，则其长度信息记录在src\_len所指向的变量中。注意：默认情况下，recvfrom是一个阻塞的调用，即直到它接收到数据才会返回。

**Close**

* 关闭socket该系统调用用来终止服务器和客户上的套接字连接，我们应该总是在连接的两端（服务器和客户）关闭套接字。

## 2.实验思路

#### 2.1流程图



#### 2.2 思路描述

本次实验，通过学习Linux shell编程，实现了一个终端版本的贪吃蛇小游戏。

* 通过w，a，s，d四个按键控制贪吃蛇的移动方向；
* 通过l键来改变贪吃蛇的速度；
* 通过 space（空格键）来暂停游戏；
* 通过n键来开始新一局游戏；
* 通过q来退出游戏，同时生成游戏报告result.txt（列举出每次游戏的分数，并给出最高分）；

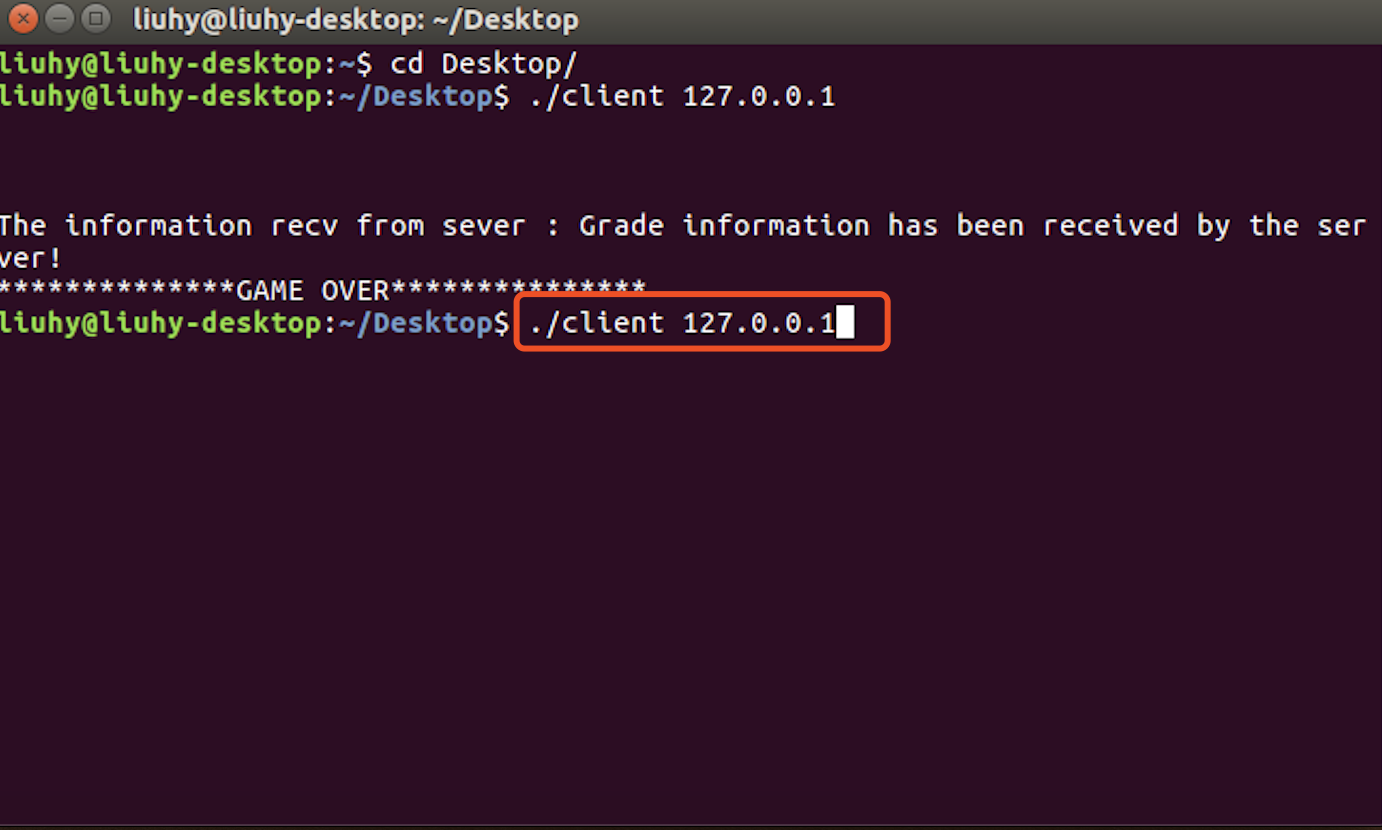
客户端通过读取result.txt中的内容，获取用户名以及本次游戏的最佳成绩，通过socket上传到服务器端，服务器接收发来的信息，生成所有玩家的排行榜。

客户端可以在成绩上传结束后，向服务器请求这个排行榜，获取所有玩家的成绩和自己的排名！

## 3. 运行界面

#### 3.1 GUI版本

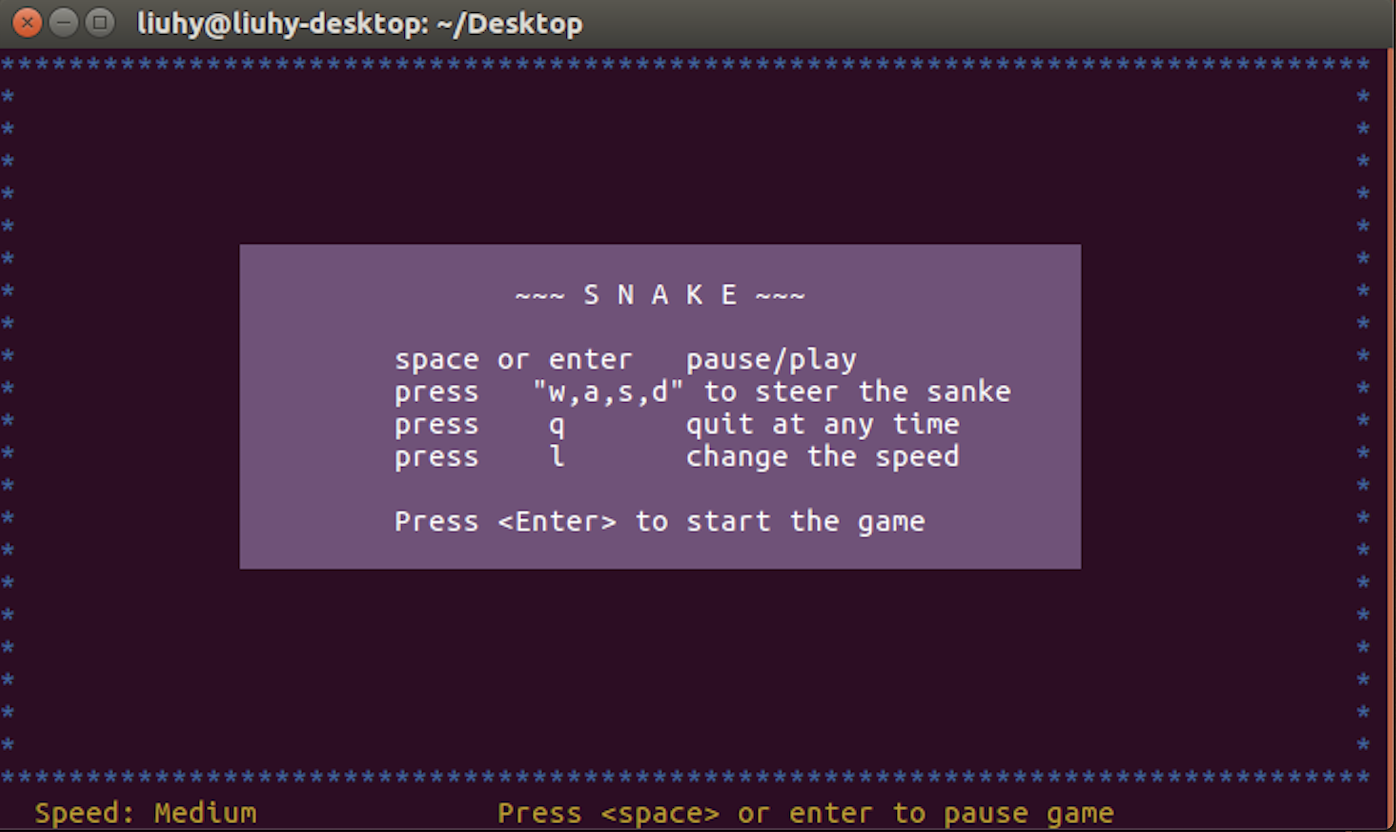
运行客户端：



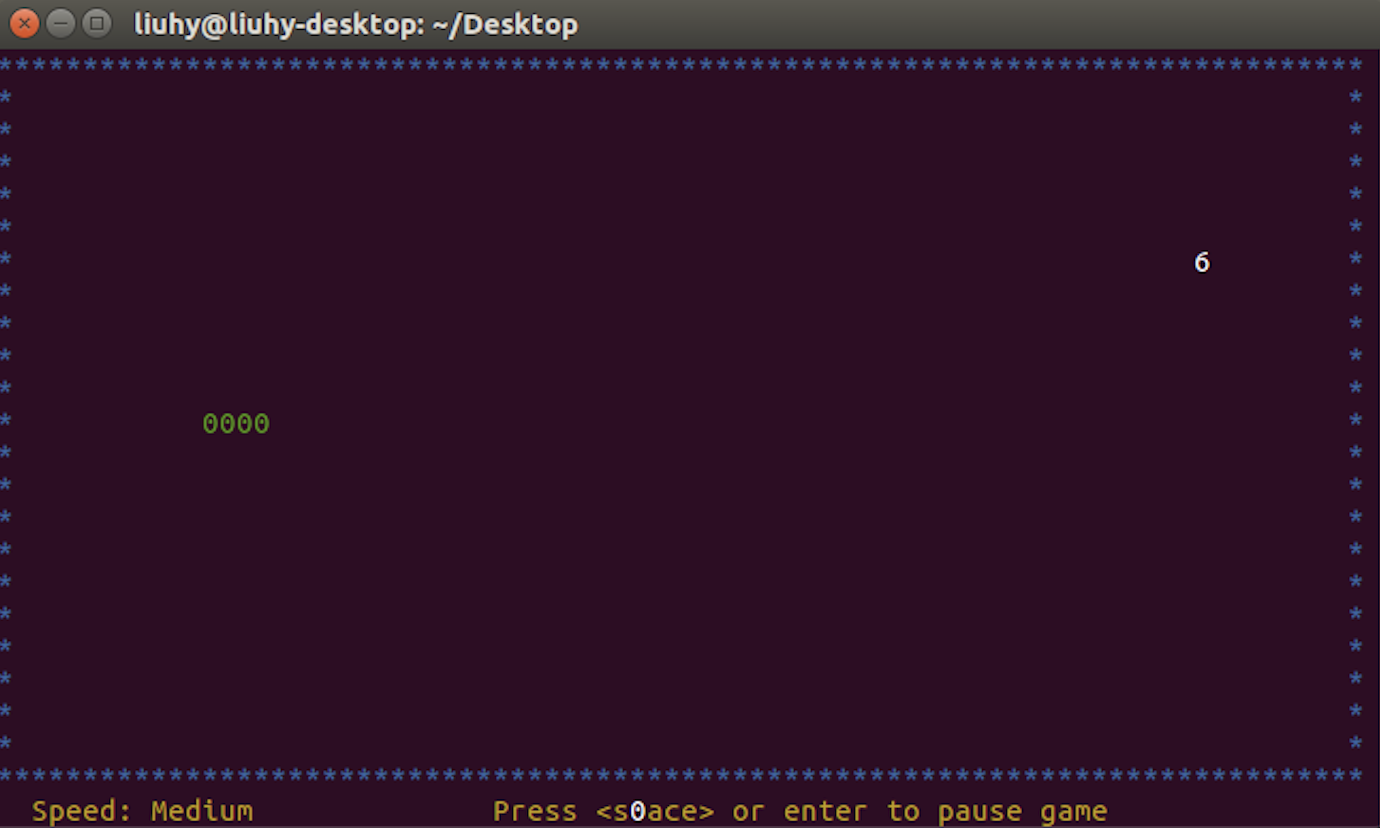
登陆界面：



输入用户名后开始游戏：



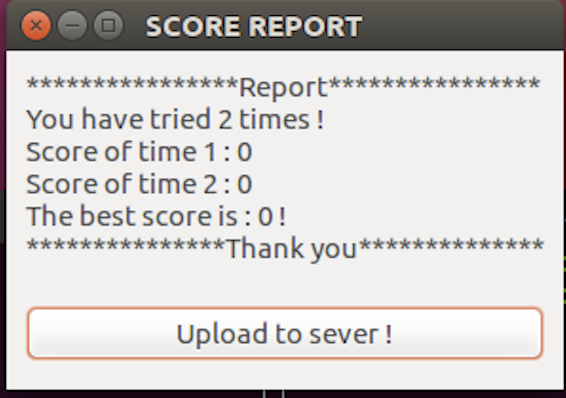
游戏开始：



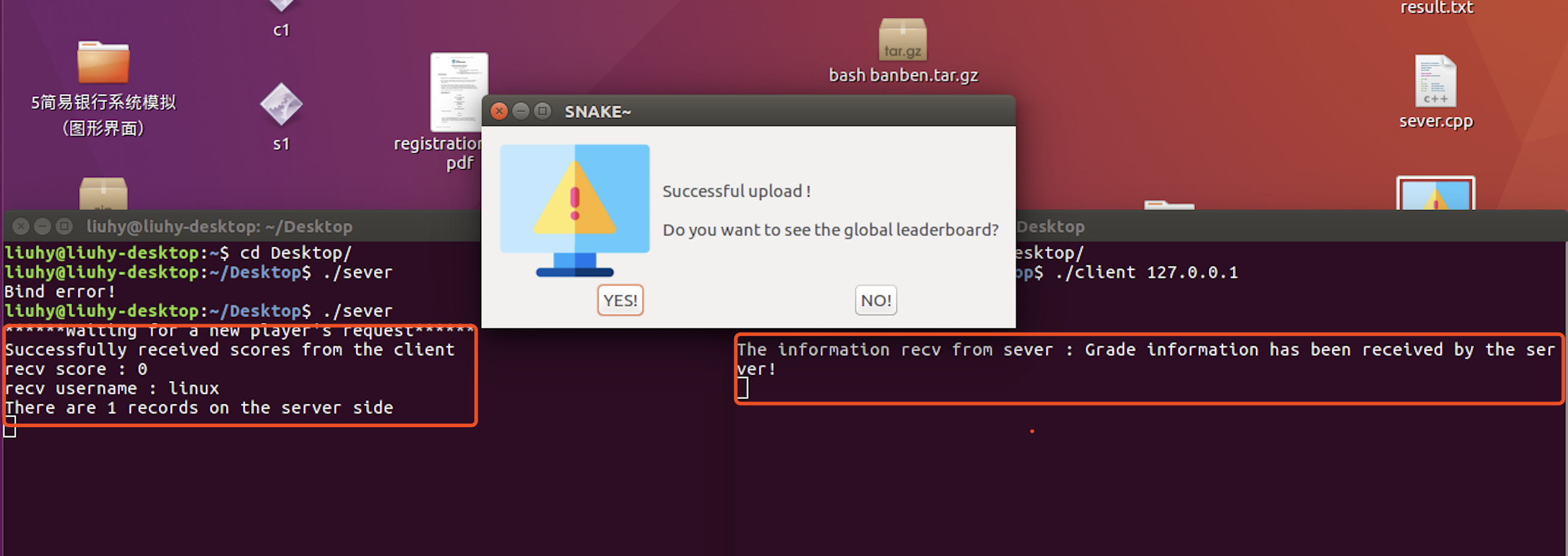
游戏结束：



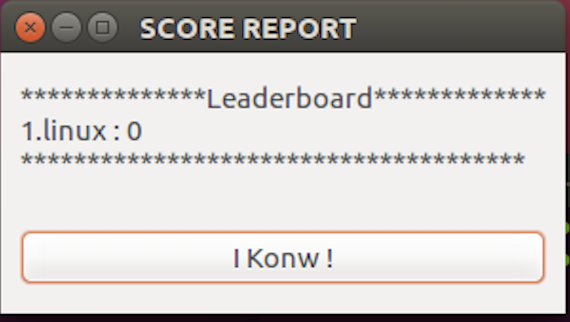
按q退出，返回成绩报告：



上传至服务器，询问是否需要查看排行榜：



单击是，获得排行榜信息：



## 实验心得

本次实验用到了文件读写、进程通信、网络编程、GTK界面编程和一些shell脚本编程的知识，实现了一个贪吃蛇游戏、其中遇见了很多困难，但是通过同学的帮助、查阅博客都能一一的解决，让自己很有成就感。通过Linux实践这门课我也学到了很多有用的知识，让我感触最深的就是socket编程。